Attorney Docket No. 15886.219
Client Docket No. 1854.CS.US.P
PATENT APPLICATION

METHOD FOR STATISTICAL SWITCHING

Inventor: Donald M. Bellenger

RELATED APPLICATIONS

This application relates to, claims the benefit of the filing date of, and incorporates by reference, the United States patent application serial number 09/089,838 entitled "Atomic Network Switch with Integrated Circuit Switch Nodes," having inventor Donald M. Bellenger, filed June 3, 1998.

U.S. PAT 6,256,306 issued 7/3/01

This application relates to, claims the benefit of the filing date of, and incorporates by reference, the United States patent number 5,802,054 entitled "Atomic Network Switch with Integrated Circuit Switch Nodes," having inventor Donald M. Bellenger, filed August 16, 1996.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to the field of network intermediate devices, and more particularly to high-performance switches for routing data in computer networks.

Description of Related Art

Network intermediate systems for interconnecting networks include various classes of devices, including bridges, routers and switches. Systems for the

interconnection of multiple networks encounter a variety of problems, including the diversity of network protocols executed in the networks to be interconnected, the high bandwidth required in order to handle the convergence of data from the interconnected networks at one place, and the complexity of the systems being designed to handle these problems. As the bandwidth of local area network protocols increases, with the development of so-called asynchronous transfer mode ("ATM"), 100 megabit per second Ethernet standards, and proposals for gigabit per second Ethernet standards, the problems encountered at network intermediate systems are being multiplied.

One technique which has been the subject of significant research for increasing the throughput of networks is known as the so-called atomic LAN. The atomic LAN is described for example in Cohen, et al., "ATOMIC: A low-cost, Very High-Speed, Local Communication Architecture", 1993 International Conference on Parallel Processing. There is a significant amount of published information about the atomic LAN technology. Felderman, et al. "ATOMIC: A High-Speed Local Communication Architecture", *Journal of High Speed Networks*, Vol. 1, 1994, pp. 1-28; Cohen, et al., "ATOMIC: A Local Communication network Created Through Repeated Application of Multicomputing Components", DARPA Contract No. DABT63-91-C-001, October 1, 1992; Cohen et al., "The Use of Message-Based Multicomputer Components to Construct Gigabyte Networks", DARPA Contract No. DABT63-91-C-001, published June 1, 1992; Finn, "An Integration of Network

Communications with Workstation Architecture", ACM, A Computer Communication Review, October 1991; Cohen et al., "ATOMIC: Low-cost, Very-High-Speed LAN", DARPA Contract No. DABT63-91-C-001 (publication date unknown, downloaded from Internet on or about May 10, 1996).

The atomic LAN is built by repeating simple four port switch integrated circuits in the end stations, based on the well known Mosaic architecture created at the California Institute of Technology. These integrated circuits at the end stations are interconnected in a mesh arrangement to produce a large pool of bandwidth that can cross many ports. The links that interconnect the switches run at 500 megabits per second. Frames are routed among the end stations of the network using a differential source route code adapted for the mesh. One or more end stations in the mesh act as "address consultants" to map the mesh and calculate source route codes. All of the links are self timed, and depend on acknowledged signal protocols to coordinate flow across the links to prevent congestion. The routing method for navigating through the mesh, known as "worm hole" routing is designed to reduce the buffering requirements at each node.

The atomic LAN has not achieved commercial application to a significant degree, with an exception possibly in connection with a supercomputer known as Paragon from Intel Corporation of Santa Clara, California. Basically it has been only a research demonstration project. Critical limitations of the design include the fact that it is based on grossly non-standard elements which make commercial use

impractical. For example, there is no way to interface the switch chips taught according to the atomic LAN project with standard workstations. Each workstation needs a special interface chip to become part of the mesh in order to participate in the LAN. Nonetheless, the ATOMIC LAN project has demonstrated a high throughput and readily extendable architecture for communicating data.

Typical switches and routers in the prior art are based on an architecture requiring a "backplane" having electrical characteristics that are superior to any of the incoming links to be switched. For example, 3Com Corporation of Santa Clara, California, produces a product known as NetBuilder2, having a core bus backplane defined which runs at 800 megabits per second. This backplane moves traffic among various local area network external ports.

There are several problems with the backplane approach typical of prior art intermediate systems. First, the backplane must be defined fast enough to handle the largest load that might occur in the intermediate system. Furthermore, the customer must pay for worst case backplane design, regardless of the customer's actual need for the worst case system. Second, the backplane itself is just another communication link. This communication link must be completely supported as a backplane for the network intermediate system, involving intricate and expensive design. The lower volumes for specialized backplane link further increases the cost of network intermediate systems based on the backplane architecture.

In light of the ever increasing complexity and bandwidth requirements of network intermediate systems in commercial settings, it is desirable to apply the atomic LAN principles in practical, easy to implement, and extendable network intermediate systems.

SUMMARY OF THE INVENTION

A method for switching packets on a network is described. The method includes computing a tag for the packet. The tag can be generated by masking portions of the packet and using the selected portions of the packet as the seed to a pseudo random number generator. The tag can then be looked up in a table. The table can be a cache of entries with one entry for each active flow. The entry is indexed by the tag. Each entry associates switching information with a tag. The switching information can be used to switch the packet.

In one embodiment, if there's no entry for the tag in the table, the packet is sent to a system for routing a packet and determining switching information.

In one embodiment, the table is updated with switching information, once the routing is complete.

In one embodiment, an entry is stored in the table for the tag indicating that other packets that generate the same tag should be dropped until the switching location is determined.

In one embodiment, the entries of the table are removed if a tag corresponding to the entry has not been looked up for a predetermined period. In one

embodiment, the predetermined period is sixty-four seconds. In another embodiment, the probability of two packets generating the same tag is used to determine the timeout period.

In one embodiment, multiple tags are generated for the packet by multiple flow detectors. In this embodiment, multiple tables can be used with a different table for the tags and switching information for the flow detectors. Alternatively, the tags can include information about the associated flow detector.

In one embodiment, the error rate is measured based on the number of matches between tags without regard to which flow detector is associated with the tag. A warning can be issued when the error rate exceeds a predetermined level. The timeout period for entries in the table can be decreased when the error rate increases above a predetermined level. The timeout period can be increased when the error rate decreases below a predetermined level.

Each of the flow detectors can operate in parallel. The pseudo random number generators used to generate the tags for all of the flow detectors can produce tags of the same length irrespective of the input lengths. Different pseudo random number generators may be used for the different flow detectors.

Sub #1

The tag can be generated from a hash code generator, a pseudo random number generator, a shift register with a feedback loop, or some other type of tag generator. The tag generator may have a non-zero probability of generating the same tag from different inputs. The length of the tag can be modified depending on the

probability of the tag generator producing the same hash code from different input packets.

In some embodiments, the flow detection is not used in conjunction with switching and routing, but instead the flow detection is used for network monitoring, billing, or other purposes. The method comprises computing a tag for a packet, looking up the tag in a table, and updating the entry associated with the tag in the table responsive to the packet. If there is no entry in the table for the tag, a new entry is created. Entries are periodically removed from the table if they have not been accessed for a predetermined period.

In one embodiment, the packet has data extracted from it to comprise the entry. For example, the source IP address can be used to determine billing information on a voice over IP call. The stored information in the table could include the number of packets. In this fashion, the usage of the network for various purposes can be determined. The packets can be sent to a system for looking up information and analyzing the packets.

In some embodiments, when the information is removed from the table, it is archived or stored in a database system or some other system for storing information.

Other aspects and advantages of the present invention can be seen upon review of the drawings, the detailed description and the claims which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified diagram of a network including an atomic network switch according to the present invention, interconnecting a plurality of standard Ethernet links.

Fig. 2 is a block diagram of a network switch based on a mesh of switch nodes according to the present invention.

Fig. 3 is a block diagram of a switch node according to the present invention.

Fig. 4 is a flow chart illustrating the process executed by the node route logic in the switch node of Fig. 3.

Fig. 5 is a diagram illustrating the process of generating identifying tags based on cyclic redundancy code hash generators for the flow detect logic of the system of Fig. 3.

Fig. 6 is a simplified block diagram of the flow detect logic for multiple parallel flows for use in the system of Fig. 3.

Fig. 7 is a flow chart illustrating the process executed in a router or other network route processor for frames received from the network switch, which do not have entries in the route tables of the network switch.

Fig. 8 is a simplified block diagram of a switch with flow detectors according to the present invention.

Fig. 9 is a flow chart illustrating the process executed in the switch with flow detectors.

DETAILED DESCRIPTION

A detailed description of embodiments of the present invention is provided with reference to Figs. 1 through 9, where Fig. 1 illustrates the context in which the present invention is utilized. In Fig. 1, an atomic network switch 10 according to the present invention is connected by standard Ethernet links 11-1 through 11-9 to a plurality of end stations 12-1 through 12-9. The number of end stations and Ethernet links shown in Fig. 1 is arbitrary. A larger or smaller number of links could be connected to a single atomic switch 10 according to the present invention, as described in detail below. Furthermore, the connections 11-1 through 11-9 from the atomic switch to the respective end stations are all standard network connections, preferably CSMA/CD protocol links, such as the standard full duplex fast Ethernet (IEEE802.3u) specified for 100 megabits per second each way, or the emerging standard full duplex, 1 gigabit per second Ethernet protocol. In the preferred system, all links 11-1 through 11-9 operate according to the same network protocol. However, alternative systems accommodate multiple network protocols on the external ports of switch 10.

The end stations 12-1 through 12-9 may be personal computers, high performance workstations, multimedia appliances, printers, network intermediate systems coupled to further networks, or other data processing devices as understood in the art.

According to one embodiment of the present invention one of the end stations, such as end station 12-1 includes resources to manage the configuration of the atomic network switch 10, such as initializing route tables, maintaining the route tables, and providing other functions. Thus, end station 12-1 may include resources to act as a multi-protocol router, such as the NetBuilder2 manufactured by 3Com Corporation of Santa Clara, California.

Fig. 2 illustrates the internal architecture of the atomic network switch 10 shown in Fig. 1. The atomic network switch 10 is comprised of a plurality of switch nodes arranged in rows and columns in Fig. 2. The switch nodes are labeled in the drawing by column and row numbers. Thus, the switch node in the upper left hand corner is node 1-1. The switch node at row 1, column 2 is node 1-2, and so on throughout the mesh. In a preferred embodiment, each switch node includes an integrated circuit, such as integrated circuit 105 in node 1-1, coupled to a memory chip, such as chip 106 in node 1-1. Each of the nodes includes four ports. Thus, node 1-1 includes port 101, port 102, port 103, and port 104.

The boundary of the network switch in Fig. 2 comprises the nodes 101 and 102 of node 1-1, port 110 of node 1-2, port 111 of node 1-3, 112 of node 1-3, port 113 of node 2-3, port 114 of node 3-3, port 115 of node 4-3, port 116 of node 4-3, port 117 of node 4-2, port 118 of node 4-1, port 119 of node 4-1, port 120 of node 3-1, and port 121 of node 2-1. Each of the ports 110-121, 101 and 102 on the boundary of the switch is connected to through a physical layer device, 121-134 to respective

physical communication media, such as fiber optic cables, twisted pair cables, wireless links, such as radio frequency or infrared channels, or other media specified according to standard local area network physical layer specifications. The connection between switch nodes, such as the connection 140 between port 141 on node 2-3 and port 142 on node 2-2, consist of medium independent interface connections which are defined for connection between MAC logic on a port, and medium dependent components for a port. However, these medium independent connections are connected from MAC logic to MAC logic directly. Preferably all the links between the ports in the network switch execute the same network protocol as the ports on the boundary of the switch. However, alternative systems support multiple protocol types at the boundary.

Management of the configuration of the network switch is accomplished in a router 150 which is connected across link 151 to the physical layer device 130 on the network switch.

The memory chips, such as chip 106 at node 1-1, in the network switch are used to store route tables, and as frame buffers used in routing of frames amongst the nodes of the switch.

In operation, the network switch receives and transmits standard LAN frames on physical interfaces 121-134. Preferably, the LAN interconnections comprise CSMA/CD LANs, such as 100 Megabit Ethernet (IEEE802.3u), or 1 gigabit Ethernet. When a standard frame enters the switch at one physical interface, it is

directed out of the switch through another physical interface as indicated by the address data carried by the frame itself. The individual nodes in the switch include a switch routing feature. Each individual node selects a port on which to transmit a received frame based upon the contents of the header of the incoming frame.

There are two internal modes for routing frames inside the switch. In the base mode, each node routes frames using a switch route header attached to the beginning of the regular LAN frame. The switch route header in one example consists of a series of bytes, each byte specifying one or more hops of the route. The top two bits in one byte specify a direction, in the next bits specify the distance. As a frame moves through each node, the header is updated until it reaches the target. Before a frame leaves the mesh, all the switch route bytes are stripped, and the frame has the same format as it had when it entered the mesh or, if required, a format adapted to the network protocol of the exit port.

The nodes of the switch, at least nodes on the boundary of the switch, also have a look up mode. When a frame enters the switch, with no source route header, the Ethernet addresses, or other fields of the control header of the frame are utilized access the route table. In preferred systems, a CRC-like checksum generator is run over the header of the frame, or over selected fields in the header. At the end of the header, the checksum, or the low order bits of the checksum, are used as a hash code to access a route table stored in the memory associated with the node. Other look up techniques could be utilized for accessing the route table in the memory. For

example, the destination address of the incoming frame could be used directly as an address in the table.

If there is an entry in the route table corresponding to the header of the frame, then the switch route data from the table is used to create a switch route header. The header is attached to the frame, and the frame is transmitted at the appropriate port. If no entry is found in the route table, then the frame is routed to a default address, such as the address of a multiprotocol router associated with the switch. The multiprotocol router at the default address also performs management functions such as reporting status, initializing the network, broadcast functions, and managing node route tables. Routing the frame to a default address alternatively involves attachment of a switch route header to direct the frame to the default address, or simply forwarding the frame at a default port in the local node, such that the next node in the mesh to receive the frame also looks it up in its own route table to determine whether the frame is recognized. Either way, the frame reaches the default address and is handled appropriately.

Flow control of the frames in the mesh, and at the boundary of the mesh, is based on the network protocol of the links, such as Ethernet. Therefore, in the preferred Ethernet example, if a port is not available in a target node due to a busy link, a collision on the link, or lack of memory space at the target node, the frame will be refused with a jam signal or a busy signal on the link. The sending node

buffers the frame, and retries the transmission later, according to the back off and retry rules of the protocol or other flow control techniques of the protocol.

The standard higher-speed Ethernet protocols include both half duplex and full duplex embodiments. The 100 Megabit per second Ethernet, defined by IEEE802.3u, clause 31 "MAC Control," defines a frame-based flow control scheme for the full duplex embodiment. Flow control slows down the aggregate rate of packets that a particular port is sending. The method used revolves around control frames distinguished by a unique multicast address and a length/type field in the packet. When a MAC port controller detects that it has received a control frame, the op-code in the control frame is sensed, and transmission of packets is controlled based on the op-code. In existing specifications, a single op-code PAUSE is defined. Thus, in response to the PAUSE op-code, transmission of packets is either enabled or disabled depending on the current state in a XON/XOFF type mechanism. Thus, this full duplex mode does not depend on the shared media, collision detect techniques of the classic CSMA/CD protocols.

All the proposed standards in the Ethernet family basically use the standard 802.3/Ethernet frame format, conformed to the 802.2 logical link control layer interface, and the 802 functional requirement document with the possible exception of Hamming distance. Also, the minimum and maximum frame size as specified by the current 802.3 standard and by the half or full duplex operational modes is different in the higher rate standards. Thus, the half and full duplex embodiments of

the 100 Megabit per second and Gigabit per second Ethernet standards are often referred to as CSMA/CD protocols, even though they may not fit completely within the classic CSMA/CD definition.

Fig. 3 is a simplified block diagram of a single node in the network switch according to the present invention. The node consists of an integrated circuit 200 comprising ports 201-1, 201-2, . . . 201-X. Each port includes the frame buffer and port management logic normally associated with standard bridges. Also, coupled to each of the ports, is a medium access control MAC unit 202-1, 202-2 1 . . . 202-X. The MAC units 202-1 to 202-X are coupled to medium independent interfaces MII 203-1, 203-2, . . . 203-X.

In the embodiment of Fig. 3, each of the medium independent interfaces is connected to a connector jack 260-1, 260-2, 260-X. The connector jacks comprise a standard connector to which a cable 270-1, 270-2, 270-X is easily connected by the user. The cable may comprise a coaxial cable for medium independent interfaces based on serial data, or ribbon cables for wider data buses. A variety of mechanical jack configurations can be used as known in the art. For example, coaxial stubs can be mounted on printed circuit boards adjacent each port of the integrated circuits. A short coaxial cable is then connected from stub-to-stub in order to arrange the plurality of integrated circuit chips in a mesh that suits the particular installation. Also, standard ribbon connector jacks can be surface mounted on printed wiring

boards adjacent to the integrated circuit. The ribbon cables are connected into the ribbon connector jacks in order to establish the inter-connection.

In alternatives, each of the switches is mounted on a daughter board, with jacks designed to be connected to a motherboard in which the data is routed according to the needs of the particular application. In alternative systems, the jacks 260-1 through 260-X are not included, and the medium independent interfaces are routed in the printed wiring board in a hard-wired configuration, designed for a particular installation.

Medium independent interfaces allow for communication by means of the jacks 260-1 to 260-X and cables 270-1 to 270-X, or otherwise, directly with other MAC units on other switch integrated circuits, or to physical layer devices for connection to actual communication media. For example, the MII 203-1 in Fig. 2 is connected directly to a port on another node in the switch. The MII 203-2 in Fig. 2 is connected to a physical layer device 204 for port 2 through jack 271. The physical layer device 204 is connected to a physical transmission medium 205 for the LAN being utilized. The MII 203-X in Fig. 2 is coupled directly to another chip within the switch mesh.

According to one embodiment of the present invention, integrated circuit 200 includes a memory interface 206 for connection directly to an external memory, such as a Rambus dynamic random access memory RDRAM 207. The RDRAM 207 is

utilized to store the switch route table 220, and for frame buffers 221 utilized during the routing of frames through the node.

The internal architecture of the integrated circuit 200 can take on a variety of formats. In one preferred embodiment, the internal architecture is based on a standard bus architecture specified for operation at 1 Gigabit per second, or higher. In one example, a 64 bit-wide bus 210 operating at 100 Megahertz is used, providing 6.4 Gigabits per second as a theoretical maximum. Even higher data rates are achievable with faster clocks. The integrated circuit of Fig. 3 includes bus 210 which is connected to a memory arbiter unit 211. Arbiter unit 211 connects the bus 210 to a CPU processor 212 across line 213. The processor 212 is utilized to execute the route logic for the node. Each of the switch ports 201-1 to 201-X is coupled to the bus 210, and thereby through the arbiter 211 to the CPU 212 and the memory interface 206. Also, flow detect logic 215 is coupled to the bus 210 for the purpose of monitoring the frame received in the node to detect flows, and to generate identifying tags for the purpose of accessing the switch route table in the RDRAM 207. The arbiter 211 provides for arbitration amongst the ports, the flow detect logic, the memory, and the CPU for access to the bus, and other management necessary to accomplish the high speed transfer data from the ports to the frame buffers and back out the port.

A representative location 250 of the switch route table is shown. The location 250 includes a field 251 for the identifying tag, a field 252 for the route header, a

field 253 for a block-unblock control bit, and a field 254 or fields for information used in the management of the route table, such as the age of the entry. The tag field 251 may be associated with a location by one or more of using the tag or a portion of the tag in the address, by storing all or part of the actual tag data in the addressed location, or by using other memory tag techniques.

The route header in the preferred embodiment consists of a sequence of route bytes. The first field in a route byte includes information identifying a direction, which corresponds to a particular port on the node, and a second field in the byte includes a count indicating the number of steps through the switch from node to node which should be executed in the direction indicated by the first field. For example, an eight bit route byte in a switch having nodes with four ports, includes a two bit direction field, and a six bit count field, specifying up to 63 hops in one of four directions. A sequence of route bytes is used to specify a route through the switch. Thus, the switch route header uses source routing techniques within the switch for the purposes of managing flow frames through the switch. The source route approach may, for example, in a 4 port node include a field for hops to right, hops to the left, hops up and hops down. The first field may carry information indicating left 4 hops, followed by a field indicating down 2 hops, followed by a field indicating left one hop to exit the switch. Thus, a frame would be transmitted out the left and in the right port of 3 nodes, in the right and out the down port of 1 node, in the top and out the down of 1 node, and in the top and out the left of the last

node on the boundary of the switch. A standard Ethernet frame format takes over for transmission through the network outside the switches. As the size of the mesh grows, and the bandwidth handled by the mesh increases, more sophisticated routing techniques are available because of the flexible technology utilized. For larger switches, more than one route exists for frames entering one node and leaving on another node. Thus, the switch can be configured to minimize the number of frames which are blocked in passage through the switch, while maintaining optimum utilization of the bandwidth available through the switch.

The block-unblock field 253 is used during the updating of the switch route table by the host CPU 212 to block routing of frames corresponding to new entries, until it is assured that the first frame in the flow to which the entry corresponds, arrives at its destination before the node begins forwarding following frames in the flow to the destination using the route header, in order to preserve the order of transmission of the frames. The age field 254 is used also by the CPU 212 for the purpose of managing the contents of the route table. Thus, entries which have not been utilized for a certain amount of time are deleted, or used according to least-recently-used techniques for the purposes of finding locations for new entries. Other control fields (not shown) include a field for storing a count of the number of packets forwarded by the node using this route, a drop/keep field to indicate packets that will be dropped during overflow conditions, a priority "high/low" field for

quality of service algorithms, and additional fields reserved for future use, to be defined according to a particular embodiment.

The frame buffer 221 is preferably large enough to hold several frames of the standard LAN format. Thus, a standard Ethernet frame may comprise 1500 bytes. Preferably, the frame buffer 221 is large enough to hold at least one frame for each of the ports on the flow switch.

The flow switch 200 includes more than 2 ports, and preferably 4 or more ports. All the ports are either connected through the media independent interfaces 203-1 through 203-X directly to other chips in the mesh, or to physical layer devices for connection to external communication media.

The router or other management node for the switch may communicate with each of the nodes 200 using well-known management protocols, such as SNMP (simple network management protocol), enhancements of SNMP, or the like. Thus, the RDRAM 207 associated with each node also stores statistics and control data used by the management process in controlling the switch node.

Although in Fig. 3, the RDRAM 207 is shown off the chip 200, alternative embodiments incorporate memory into the switch integrated circuit 200, for more integrated design, smaller footprint for the switch, and other classic purposes for higher integration designs.

The CPU 212 executes the node route logic for the node. A simplified flow chart of the node route process executed by CPU 211 is shown in Fig. 4.

The process begins with the receipt of the frame on a particular port (step 300). The CPU first determines whether the frame carries a route header (step 301). This process is executed in parallel with the transferring of the frame being received to the frame buffer of the node. If the frame carries a route header, then the CPU updates the header by decrementing the hop count, or otherwise updating the information to account for a traversed leg of the route according to the particular switch route technique utilized. The CPU transmits the frame (with updated header) on the port identified by the header (step 302). If at step 301, no switch route header was detected, the flow detect logic is accessed to determine a tag for the frame (step 303). The tag is utilized by the CPU to access entries in the route table (step 304). If a match is found in the route table, then a route header is generated for the frame (step 305). Then, the header is updated (if required), and the frame is transmitted on the port identified by the data in the table (step 302). If at step 304, no match was found in the route table, then the frame is transmitted on a default port (step 306). An alternative technique to transmitting the frame on a default port, is to add a default route header to the frame, and transmit the frame according to the information in the default route header. In this manner, subsequent nodes in the switch will not be required to perform the look-up operation for the purposes of routing the frame. However, it may be desirable to have each node look up the frame in its own route table, in order to insure that if any node already has data useful in

forwarding the frame, then that frame will be forwarded appropriately without requiring processing resources of the management process at the default address.

Fig. 5 illustrates the technique executed by the flow detect logic in generating an identifying tag for the frame being received. Fig. 5 includes the format of a standard Ethernet (802.3) style frame 400. The frame includes a start of frame deliminator SOF in field 401. A destination address is carried in field 402. A source address is carried in field 403, and miscellaneous control information is carried in additional fields 404. A network layer header, such as an Internet protocol header in this example, is found in field 405. Other style network layer headers could be used depending on the particular frame format. The data field of variable length is found at section 406 of the frame. The end of the frame includes a CRC-type checksum field 407 and an end-of-frame deliminator 408. The flow detect logic runs a CRC-type hash algorithm over selected fields in the control header of the frame to generate a pseudo random tag. Thus, the field 410, the field 411, the field 412, and the field 413 are selected for input into a CRC hash generator 414. The tag generated by the hash generator 414 is supplied on line 415 for use in accessing the route table 416. The route table either supplies a route header on line 417, or indicates a miss on line 418. In this way, the route management software executed by the CPU can make the appropriate decisions.

The embodiment of Fig. 5 selects a particular set of fields within the frame for the purpose of generating the pseudo random tag. The particular set of fields is

selected to correspond to one standard frame format encountered in the network. However, a variety of frame formats may be transmitted within a single Ethernet style of network, although in this example, a CRC-type hash generator is utilized, relying on typical CRC-type algorithms, referred to as polynomial arithmetic, modulo II. This type of arithmetic is also referred to as "binary arithmetic with no carry" or serial shift exclusive-OR feedback. However, a variety of pseudo random number generation techniques can be utilized, other than CRC-like algorithms. The two primary aspects needed for a suitable pseudo random hash code are width and chaos, where width is the number of bits in the hash code, which is critical to prevent errors caused by the occurrence of packets which are unrelated but nonetheless result in the same hash being generated, and chaos is based on the ability to produce a number in the hash register that is unrelated to previous values.

Also, according to the present invention, the parsing of the frames incoming for the purposes of producing an address to the look-up table can take other approaches. This parsing can be referred to as circuit identification, because it is intended to generate a number that is unique to the particular path of the incoming frame.

The circuit identification method depends on verifying a match on specific fields of numbers in the incoming frame. There are two common table look-up methods, referred to as binary search and hash coding. The key characteristic of binary search is that the time to locate an entry is proportional to the log base 2 of

the number of entries in the table. This look-up time is independent of the number of bits in the comparison, and the time to locate a number is relatively precisely known.

A second, more preferred, method of look-up is based on hash coding. In this technique, a subset of address field or other control fields of the frame are used as a short address to look into the circuit table. If the circuit table contains a match to the rest of the address field, then the circuit has been found. If the table contains a null value, then the address is known not to exist in the table. The hash method has several disadvantages. It requires a mostly empty table to be efficient. The time to find a circuit cannot be guaranteed. The distribution of duplicates may not be uniform, depending on the details of which fields are selected for the initial address generation.

The address degeneracy problem of the hash coding technique is reduced by processing the initial address fragment through a polynomial shift register. This translates the initial address to a uniformly-distributed random number. A typical example of random number generation is the CRC algorithm mentioned above. In a preferred hashing technique, the hardware on the flow switch includes at least a template register, pseudo random number generation logic and a pseudo random result register. The template register is loaded to specify bytes of a subject frame to be included in the hash code. The template specifies all protocol-dependent fields for a particular protocol. The fields are not distinguished beyond whether they are included in the hash or not. As the frame is processed, each byte of the initial header

is either included in the hash function or it is ignored, based on the template. A hash function is generated based on the incoming packet and the template. The pseudo random number generator is seeded by the input hash bits selected by the template. The change of a single bit in the input stream should cause a completely unrelated random number to be generated. Most common algorithms for generating pseudo random numbers are linear-congruential, and polynomial shift methods known in the art. Of course, other pseudo random number generation techniques are available.

A first field of the pseudo random number is used as an address for the look-up table. The number of bits in this field depends on the dimensions of the look-up table. For example, if the circuit table has 64,000 possible entries, and the hash number is eight bytes long, the first two bytes are used as an address. The other six bytes are stored as a key in the hash table. If the key in the hash table matches the key in the hash code, then the circuit is identified. The additional bytes in the table for the addressed entry specify the route to be applied. The length of the pseudo random hash code is important, to account for the probability that two unrelated frames will result in the same hash number being generated. The required length depends on the size of the routing tables, and the rate of turnover of routes.

The problem with a pure hash code circuit identification technique is that there is a chance of randomly misrouting a packet. The problem arises when you are generating random numbers out of a larger set. There is a chance that two different input patterns will produce the same hash code. Typically, a hash code will be

loaded into a table with a known route. Then a second, different, packet will appear that reduces to the same hash code as the one already in the table. The second packet will be falsely identified as having a known route, and will be sent to the wrong address.

This error can be understood by the well-known statistics of the "birthday problem." The "birthday problem" answers the question, "What is the probability that two people in a group will have the same birthday?" It turns out that the number of people in a group required for there to be a likelihood of two people having the same birthday is quite small. For example, there is a 50% chance that two people out of a group of 23 will have the same birthday.

The probability of a switching error depends on the number of circuits active. For example, if there are no circuits active, then there is no chance that an invalid circuit will be confused with another circuit, since there are no valid circuits. As each circuit is added to the table, it decreases the remaining available space for other numbers by approximately $(1/2)^{bits}$, where "bits" is the number of bits in the hash code. If the hash code is 32 bits long, then each entry into the circuit table will reduce the remaining code space by $(1/2)^{32}$, which is equal to 2.32×10^{-10}. The cumulative probability of not making an error in the circuit table is equal to the product of the individual entry errors up to the size of the table. This is $(1)*(1-1/2^{32})*(1-2/2^{32})*(1-3/2^{32}) \ldots *(1-n/2^{32})$, where n is the number of entries in the table. In the case of a 32-bit hash code, and an 8,000-entry circuit table, the probability of

making an error in the table would be about 0.7%. With a 64,000-entry circuit table, the probability of an error would be about 39%.

Using a 32-bit hash code and some typical-sized circuit tables indicates that the conventional wisdom is correct. That is, there will be routing errors if only a 32 bit hash code is used. However, if the number of bits in the hash code is increased and probability is recalculated for typical-sized circuit tables, we find that the probability of error quickly approaches zero for hash codes just slightly longer than 32 bits. For example, an 8,000-entry table with a 40-bit hash code will reduce the error rate to 0.003%. A 48-bit hash code will reduce the error to 0.000012%. These calculations show that a pure hash code look-up table can be used if the length of the hash code is longer than 32 bits for typical-size tables.

As a further example, consider the case of a 64-bit hash code. Assuming an 8,000-entry table, the probability of making an error is $2*10^{-12}$. Even if the table is completely replaced with new entries every 24 hours, it would take over one billion years for an error to occur. Using a 64-bit hash code with a 64,000 entry table would give a probability of error of 10^{-10}. Assuming the table turned over every day, it would take about 28 million years for an error to occur. An error might occur sooner, but the rate would be negligible. In all cases, there is no realistic chance of making an error based on this routing technique within the lifetime of typical networking equipment.

The "birthday problem" analysis of the circuit cache can underestimate the error rate of a statistical switch. The analysis correctly computes the probabilities of errors as the cache is initially loaded; however, more generally the error rate can be slightly higher. A true pseudo random number generator will produce a uniform distribution. For example, with an n bit pseudo random number generator, any of the 2^n possible outputs should be equally likely. One advantage of a pseudo random number generator for this application is that for the same input, or seed, the same output is always produced.

Nonetheless, if the input space is smaller than the output space of the pseudo random number generator, there is an obvious problem. If there is an 8 bit input space and an 64 bit pseudo random number generator, only 256 of the 2^{64} possible outputs can be generated, but perhaps fewer because perhaps two or more of the inputs might generate the same output. Extending this example, consider the case where there are only 256 possible input flows, or circuits, and 255 of the flows are already active in the cache. When a packet arrives that matches the last uncached flow, the probability that the new flow will be misidentified is equal to the number of active entries divided by the output space of the pseudo random number generator, or $\frac{2^8 - 1}{2^{64}}$. This is because of the 2^{64} possible results, only 255 are "bad". In typical embodiments, there will be approximately 10^6 flows active in the cache at a given time. Regardless of the size or length of the flow, each flow will be set up

for switching. No analysis is done of short flows or long flows. The flows, or circuits, are automatically timed out after a predetermined idle period. In one embodiment, each flow is left in the cache for sixty-four (64) seconds. In another embodiment, the time out is adjusted based on the measured potential error rate by using multiple flow detectors. The error does not vary with the table size if the table is organized like a cache so that an initial portion of the tag is used to designate another section of memory to search on the remainder of the tag.

An optical carrier 3 (OC-3) channel is used for many points of the Internet backbone. By monitoring an OC-3 channel, an average number of flows was computed as 356,735. Further, each second, there were 4,594 flows started. A flow on the network was characterized by having the same protocol, source address, destination address, source port and destination port, with a sixty-four second time out. Given these observations and a 64 bit pseudo random number generator, there are 356,735 chances out of 2^{64} to make an error for each new flow added to the cache and the probability of misrouting a new flow is approximately 1.93×10^{-14}. Using this data, it can also be determined that an OC-3 statistical switch will make one error approximately every 350 years, or $4{,}594 \times (1.93 \times 10^{-14}) = 8.88 \times 10^{-11}$ errors per second. Also, handling a packet setup rate of 4,594 packets per second is well within the capabilities of a most routers. However, each second 35,607 packets will be switched on average and 83,231,481 bits per second are switched. These numbers are for a half duplex flow, a full duplex four port switch would make four

times as many errors. To support larger OC connections, the results scale linearly, OC-12 would require a circuit table for caching 1,426,940 active flows and OC-48 would require four times that. The routing requirements for an OC-12 would be approximately 20,000 packets per second, and approximately 80,000 packets per second for OC-48. The OC-12 statistical switch would make one error every 89 years and the OC-48 statistical switch would make one error every 22 years.

The error rate can be made observable and tested by reducing the number of bits generated by the pseudo random number generator. By reducing the tag length to 32 bits and then increasing the tag length a bit at a time, the error rate can be directly observed. With each added bit, the error rate should drop by a factor of 2, and this is what can be verified.

In a preferred embodiment, filtering mechanisms are implemented on the flow switch, and multiple filters operate in parallel. The circuit look-up table can be implemented with external memory much larger than the number of circuits expected to be simultaneously active. This means that the hash pointer generated either points to a valid key or a miss is assumed. There is no linear search for matching key. In another embodiment, an associative memory (CAM) is used to provide the lookup functionality. In this embodiment, a portion of the key is used to look up entries in the CAM. The matching entries can then be searched such as by a linear search. Other table arrangements are possible.

When a circuit is not found in the table, the packet is routed to a default address. Normally, this default address directs the packet to a stored program router. The router will then parse the packet using standard methods, and then communicate with the flow switch circuit to update the circuit table with the correct entry. All subsequent packets are directly routed by the switch element without further assistance from the router.

Example template organizations for the bridging embodiment, the IP routing embodiment, and the IPX routing embodiment are set forth below. Example for bridging:

Basic Ethernet packet:	Preamble 64 bits are discarded	
Destination Address:	bytes 1-6	Used
Source Address:	bytes 7-12	Used
Packet Type:	bytes 13-14	are ignored (802.3 length)
Data bytes:	15 up to 60	are ignored
CRC:	Last 4 bytes	are ignored

The template register is 8 bytes long. Each bit specifies one byte of the header. The first bit corresponds to byte 1 of the Destination Address.

The template for bridging is FF-F0-00-00 00-00-00-00

The selector is: Always TRUE. Hierarchy=1 (default to bridging)

Example for IP:

Preamble 64 bits are discarded		
Destination	bytes 1-6	optional
Source	bytes 7-12	optional
Packet type	bytes 13-14	Ignore (802.3 length)
byte 15:	IP byte 1	= version length = optional
byte 16:	IP byte 2	= service type = Ignore
17 - 18:	IP 3 - 4	= length = Ignore
19 - 22:	IP 5 - 8	= Ignore
23	IP 9	= TTL = optional
24	IP 10	= Proto = optional
25 - 26	IP 11 - 12	= Hdr chksum = Ignore
27 - 30	IP 13 - 16	= Source IP address = Used
31 - 34	IP 17 - 20	= Destination IP address = Used
35 -	IP 21 -	= Ignore

Assume that optional fields are included in the pseudo random hash code.

The template would then be: FF-F2-03-03 FC-00-00-00

The selector is: Bytes 13-15 = 080045, Hierarchy = 2

Example for IPX in an Ethernet frame:

Preamble 64 bits are discarded			
Destination	bytes 1 - 6	Optional	
Source	bytes 7 - 12	Optional	
Type	bytes 13 - 14	Optional (Selector = 8137)	
byte	IPX		
15-16	1 - 2	Checksum	Ignore
17 - 18	3 - 4	Length	Ignore
19	5	Hop count	Optional
20	6	Type	Optional (Selector = 2 or 4)
21 – 24	7 - 10	Dest Net	Use
25 - 30	11 – 16	Dest Host	Use
31 – 32	17 – 18	Dest Socket	Ignore
33 - 36	19 - 22	Src Net	Use
37 - 42	23 - 28	Src Host	Use
43 -	29 -		Ignore
Template (with optional fields):			FF-FC-3F-FC FF-CO-00-00
Selector:	Bytes 13 - 14 = 8137, Hierarchy = 2		

The examples shown are representative, and may not correspond to what would actually be required for any particular application. There are many protocol

pattern possibilities. Some combinations may not be resolvable with the hierarchy described in these three examples.

In the embodiment in which there are a number of filters operating in parallel, the flow detect logic includes the template register discussed above, a second register loaded with a template for detecting the specific protocol type represented by the template register. This feeds combinational logic that provides a boolean function, returning a true or false condition based on a string compare of a section of the frame to determine the protocol. A third register is loaded with a hierarchy number, which is used to arbitrate among similar protocols, which might simultaneously appear to be true based on the second protocol detect register. A fourth register is optional, and contains a memory start address which triggers the operation of the filter.

The multiple instantiations of the filters operate in parallel. The filters can be reprogrammed on the fly to support the exact types of traffic encountered. Furthermore, the filters may operate in a pipeline mode along a series of switching nodes. Each protocol returns its hierarchy number when that filter detects the protocol pattern contained in the template. For example, bridging protocol may be defined as true for hierarchy 1 for all frames. If no stronger filter fires, such as an IP or IPX filter, then the bridging filter will be selected as the default.

Thus, the flow detect logic in a preferred system executes a plurality of hash flow analyses in parallel as illustrated by Fig. 6. Thus in Fig. 6, a received frame is

supplied on line 500 in parallel to hash flow logic 1 through hash flow logic N, each flow corresponding to a particular frame format. Also, the received frame is supplied to a hash flow "select" 501 which is used for selecting one of the N flows. The output of flows 1 through N are supplied through multiplexer 502 in Fig. 6, which is controlled by the output of the select flow 501. The output of the select flow 501 causes selection of a single flow on line 503, which is used for accessing the route table by the CPU.

Thus a preferred embodiment of the present invention uses a routing technique base on flow signatures. Individual frames of data move from one of the Ethernet ports to a shared buffer memory at the node. As the data is being moved from the input port to the buffer, a series of hash codes is computed for various sections of the input data stream. Which bits are or are not included in each hash calculation is determined by a stored vector in a vector register corresponding to that calculation. For example, in the most common case of an IP packet, the hash function starts at the 96th bit to find the "0800" code following the link-layer source address, it then includes the "45" code, 32 bits of IP source, 32 bits of IP destination, skips to protocol ID 8 bits, and then at byte 20 takes the source port 16 bits and the destination port 16 bits. The result is a 64 bit random number identifying this particular IP flow.

The hash code is looked up in or used to access a local memory. If the code is found, it means that this flow type has been analyzed previously, and the node will

know to apply the same routing as applied to the rest of the flow. If there is no entry corresponding to this hash code, it means that the flow has not been seen lately, and the node will route the frame to a default destination. A least recently used algorithm, or other cache replacement scheme, is used to age flow entries out of the local tables.

In practice, many filters operate simultaneously. For example, filters may be defined for basic bridging, EP routing, sub-variants, Apple Talk, and so on. The actual limit to the number of filters will be determined by the available space on the ASIC. The logic of the filters is basically the same for all the filters. The actual function of each filter is defined by a vector register specifying which bits are detected.

A second feature is the use of multi-level filters. In the common case simultaneously supporting bridging, IP, and IPX; about ten filters operate in parallel. An additional level of coding is used to select which of the other filters is to be used as the relevant hash code. This second level filter would detect whether the flow was IP or IPX for example.

In the case where the flow is not recognized, it is passed to the default route. As the packet passes along the default route, additional nodes may examine the packet and detect its flow type based on different filters or on a different set of flow signatures (hash table entries) stored. This method of cascading filters and tables allows for the total size and speed of the mesh to be expanded by adding nodes.

Ultimately, if a packet can not be routed by any of the nodes along the default route, the packet will arrive at the final default router, typically a NetBuilder2. The default router will analyze the packet using standard parsing methods to determine its correct destination. A flow signature will be installed in an appropriate node, or nodes, of the mesh so that subsequent flows of the same signature can be routed autonomously without further intervention.

A flow effectively defines a "circuit" or a "connection"; however, in standard Ethernet design, packets are treated individually without any regard to a connection. Typically a router will analyze every single packet as if it had never seen it before, even though the router might have just processed thousands of identical packets. This is obviously a huge waste of routing resources. The automation of this flow analysis with multiple levels of parallel and cascaded hashing algorithms combined with a default router is believed to be a significant improvement over existing routing methods.

Flow based switching is also critical to ensuring quality of service guarantees for different classes of traffic.

Fig. 7 is a flow chart illustrating the process executed in the router or other management node, whenever a frame is received which does not have a switch route header. Thus, the process of Fig. 7 begins at step 700 where a frame is received in the router, such as the router 150 in Fig. 2. The router applies the multiprotocol routing techniques to determine the destination of the frame. Based on the

destination, and other information about the flows within the switch, switch route headers are generated for nodes in the switch (step 701). Thus, a different route header is generated for each node in the switch mesh, and correlated with the tag which would be generated according to the received frame at each node. Next, a message is sent to the nodes in the switch to update the route tables with the new route headers, and to block frames which match the tag of the frame being routed (block 702).

After step 702, the frame is forwarded from the router to its destination (step 703). After the frame has been forwarded to its destination, the router sends a message to all of the nodes in the switch to unblock frames which have a matching tag (step 704). This blocking and unblocking protocol is used to preserve the order in which frames are transmitted through the switch, by making sure that the first frame of a single flow arrives at its destination ahead of following frames.

Logic in the nodes for the purpose of accomplishing the blocking and unblocking operation take a variety of formats. In one example, the entry at each location in the route table includes a field which indicates whether the flow is blocked or not. When an entry is first made in the route table, the blocking field is set. Only after a special instruction is received to unblock the location, is the blocking field cleared, and use of the location allowed at the switch node.

Accordingly, in the preferred system the atomic network switch according to the present invention is based on repeated use of a simple 4-port switch integrated

circuit. The integrated circuits are interconnected to create a mesh with a large pool of bandwidth across many ports. The links that interconnect the integrated circuits run according to a LAN protocol, at preferably 100 megabits per second or higher, such as a gigabit per second. Individual ports act as autonomous routers between the boundaries of the switch according to the switch route protocol which is layered on top of the standard frame format. The overall bandwidth of the switch can be arbitrarily increased by adding more atomic nodes to the switch. Using a well-understood and simple interface based on standard Ethernet LAN protocols, vastly simplifies the implementation of each node in the switch, because each is able to rely on well understood MAC logic units and port structures, rather than proprietary complex systems of prior atomic LANs. Furthermore, any node of any switch can be connected to a physical layer device that connects to an Ethernet medium, or can be disconnected from the Ethernet medium and connected to another node switch to readily expand and change the topology of the switch. The fine granularity and scalability of the mesh architecture, combined with the ability to optimize the topology of the switch for a particular environment allow implementation of a high bandwidth, low cost network switch.

Fig. 8 is a simplified block diagram of a switch with flow detectors according to the present invention. The switch 802 could be an atomic network switch 10, or some other sort of statistical switch. The switch 802 is coupled to a router 808. The router 808 could be the router 150, or some other sort of router. The router 808 can

provide configuration information to the switch 802. The switch 802 has an input 800 and outputs 810-814. The input 800 is also coupled to one or more flow detectors 806-1 to 806-2. The flow detectors 806-1 to 806-2 generate tags by a quick analysis of the frames, or packets, flowing into the switch 802 over the input 800. The flow detectors 806-1 to 806-2 can use a template, or mask, to select the relevant bytes of incoming frames to use in generating a tag using a hash generator such as the hash generator 414 (see FIG. 5).

The flow detectors 806-1 to 806-2 are coupled to a cache 804. The cache is a lookup table with entries indexed by tags such as the route table 416, or some other type of table. The cache 804 can be a single cache for all of the flow detectors 806-1 to 806-2 or it can be multiple caches with separate caches for each of the flow detectors 806-1 to 806-2. For example, the flow detector 806-1 could be a hypertext transfer protocol (HTTP) flow detector and the flow detector 806-2 is a real-time transfer protocol (RTP) flow detector. Examples of RTP flows are voice over IP calls (H.323 standard), streaming audio, streaming video, and other real time flows. Because of their time sensitive nature, it may be important to provide priority to RTP flows over non-real time flows such as HTTP packets.

If a single cache is used for all of the flow detectors, the table entries should be marked to indicate which flow detector generated the tag. This is useful for reducing errors and for measuring the error rate. The switch 802 is a statistical switch and so there is a non-zero probability that a frame, or packet, will be

misdirected. By marking the entries it is easy to avoid any mistakes where an HTTP packet matches a tag for a RTP flow. Thus, although there could be twenty different flow detectors operating on the same switch, the error rate will remain independent of the number of flow detectors. The error rate of the switch can be measured in real time as well. By recording the cross-flow detector hits in the table, the error rate can be computed. If there are *n* flow detectors, then the cross-flow detector hits will be *n* times the error rate. Thus, the error rate of the switch can be directly measured.

As the packet, or frame, is carried into the switch 802 over input 800, the templates in the flow detectors 806-1 to 806-2 receive the packet and begin generating the tags. When the last required bit of a packet passes the flow detector 806-1, the tag is computed. Similarly, when the last required bit of a packet passes the flow detector 806-2, the tag is computed. Because shift registers with a feedback loop can be used to generate the pseudo random hash codes, or tags, there is only a very short delay. Because the flow detectors are coupled to the input 800 in parallel, they can operate independently. Once the tag is computed, the flow detectors 806-1 to 806-2 look up the tag in the cache 804. If an entry is found matching a tag, the flow detector will provide switching information stored with the entry to the switch 802 over the hit input 818 which is coupled to the switch.

If none of the flow detectors 806-1 to 806-2 detect a match, an entry is made in the table with the switching information to drop packets. Consider the following example, a new flow of HTTP packets is starting with a first packet X. The packet X

is part of a new flow and therefore, there is no tag in the table for either of the flow detectors 806-1 to 806-2. Two entries are made, one for each flow detector, in the table indicating to drop matching packets. When no switching instructions are provided over hit input 812 to the switch 802, the switch sends the packet to a router or other device for determining the routing information. The router 808 is capable of examining a packet and providing a switch configuration to support the transmission of the packet from the switch 802 over one or more of the outputs 810-814. The router 808 also provides training inputs 816 to the flow detectors 806-1 to 806-2.

If a second packet, packet Y, in the flow of HTTP packets arrives while the router 808 is still processing packet X, the drop entries will cause the switch to drop the packet. This avoids the need to cache the packet Y or call on the router to route packet Y while the router is still processing packet X. The TCP slow start means that flows that are transmitted using TCP will exhibit slow starts and it is unlikely that there will be many dropped packets. When the user datagram protocol (UDP) is used for video and audio, it is typically because it is undesirable to recover lost packets, therefore the dropped packets will not cause a problem.

Once the router 808 has processed packet X, the switching information is provided both to the switch 802 and to the flow detectors 806-1 to 806-2. The information can be provided over the training inputs 816. The router 808 will also indicate which of the flow detectors 806-1 to 806-2 is the correct protocol for the flow. Because packet X was part of an HTTP flow, the flow detector 806-1 which is

an HTTP flow detector will be trained and the tag entry in the cache 804 will be updated with the switching information. The information stored by the other flow detectors can either be explicitly purged at this point or the entries can be allowed to idle until they are removed automatically.

In order to control the error rate and limit the size of the cache, tags are only kept in the cache 804 if they are active. An idle time out can be selected and adjusted for the switch. For example, if the application the switch is being used for is characterized by long idle periods, it may be necessary to increase the idle time out to accommodate the application. However, if the error rate as measured by the cross-flow detector hit rate is rising, it may be desirable to shorten the idle period to reduce the number of active flows and thus reduce the number of tags in the cache.

Continuing with the example, now that the packet X has been routed and the switching information stored in the cache 804, if a third packet, packet Z, arrives before the flow is timed out as inactive, the flow detector 806-1 will generate a hit and provide the switching information from the cache 804 to the switch 802 over the hit input 818.

It is also possible to use the flow detectors 806-1 to 806-2 in other networking applications. For example, network monitoring without switching or routing on a per flow basis can be carried out. The router 808 can be replaced with a computer that provides other information from analyzing a packet. For example, billing information for a Voice Over IP call. The cache 804 can then be used to store

usage information, e.g. increment the entry for a tag for every packet. Although errors are possible, the error rate is very low and can be controlled as described. Before a cache entry is timed out, the information can be transferred to a billing database. A similar configuration can be used in network monitoring applications to show the number of active sessions and their types.

Fig. 9 is a flow chart illustrating the process executed in the switch with flow detectors. At step 900, the process starts as a packet is received. At step 902, as the packet is received the tags are generated by the flow detectors in parallel.

At step 904, if one of the flow detectors finds a match, control proceeds at step 912. If there are no matches, then a new flow has been detected and control proceeds at step 906.

At step 906, the tag is marked as drop for all of the flow detectors. This prevents subsequent matching packets in the flow from being switched or routed until the current packet is routed. Control then proceeds to step 908, where a router or other device translates the packet to provide a destination address and compute switching information. The packet is then transmitted either by the router or by the switch. Control then proceeds to step 910, where the tag table is updated to provide the switching information. The correct flow detector for the protocol stores the switching information with the tag in the tag table. The other flow detectors can eliminate their drop entries or allow the drop entries to time out. The process then ends.

At step 912, the matching tag from step 904 is examined. If the tag indicates that the packet should be dropped, the packet is dropped and the process ends. Otherwise, control proceeds at step 914. At step 914, the switching information from the matching tag is provided to the switch. At step 916, the packet is switched according to the instructions.

A high bandwidth and very flexible network switch is achievable according to the present invention with a simple, scalable, low-cost architecture.

The foregoing description of a preferred embodiment of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to practitioners skilled in this art. It is intended that the scope of the invention be defined by the following claims and their equivalents.